

**SYSTEM ARCHITECTURE AND METHOD FOR SYNCHRONIZATION OF  
REAL-TIME CLOCKS IN A DOCUMENT PROCESSING SYSTEM**

**Background of the Invention**

5 The present invention relates to a system architecture for a document processing system. It finds particular application in conjunction with synchronization of a real-time clock in one or more modules of a multiple feeding and finishing architecture with a master real-time clock in a controller of the document processing system and will be described with particular reference thereto. However, it is to be appreciated that the present invention is also amenable to other like applications.

10 In the early days of copiers, copies were fed to a tray where the operator took them and performed manual finishing operations. These manual operations ranged from simple collation and stapling to more sophisticated operations such as folding and inserting in an envelope for mailing. Later products incorporated finishing functionality within the print engine to relieve the operator of these manual finishing functions. Finishing operations included collation, stapling, and binding. The same revolution has also occurred in feeding applications. Operators originally placed the document to be  
15 copied on the platen, closed the cover and pushed the start print button. Now, automatic document feeders and recirculating document handlers have become commonplace.

Incorporating feeding and finishing functions within the print engine was well received when collation and stapling were the primary operations. However,  
20 customers soon required more and more sophisticated feeding and finishing operations. It became increasingly inefficient to follow a course of developing feeding and finishing functions specific to a single product. Such product development was a very time consuming activity, required enormous resources, and led to duplication of activities. Another issue was the large number of third party vendors with experience in the  
25 feeding and finishing technologies. It was a waste of resources to duplicate the off-line feeding and finishing functions currently provided by third party vendors. Under these

circumstances, a standard for attaching feeding and finishing devices to print engines was developed by Xerox Corporation ("Xerox"). The standard was used by Xerox for developing new document processing products and by third party vendors for developing feeding and finishing modules. The third party feeding and finishing modules were compatible with Xerox's print engines and other document processing products.

Certain aspects of the standard are described in U.S. Patent No. 5,629,775 to Platteter et al. ("Platteter '775") and assigned to Xerox, entitled "System Architecture for Attaching and Controlling Multiple Feeding and Finishing Devices to a Reproduction Machine." The basic modular architecture (i.e., multiple feeding and finishing architecture (MFFA)) of an electronic image processing apparatus in Platteter '775 inherently allowed the duplication of effort for developing feeding and finishing capabilities to be avoided by permitting and encouraging third party vendors to develop or adapt feeding and finishing devices that were compatible with standard print engines. More specifically, Platteter '775 permitted the document feeding and/or finishing modules to be interconnected with the print engine in a networked environment and organized in a sequence that meets the customer's needs for a fully automated job. This permitted finishing or feeder devices to be attached in any order that the operator decided was appropriate for the application.

However, feeding and finishing modules in the networked MFFA include clocks that must be accurately synchronized with a master clock in the master module, the print engine, in order to correctly perform the printing function. This is because the master module controls all the scheduling between the feeding and finishing modules.

As document processing systems became more sophisticated the accuracy and precision of the synchronization have become more and more important.

Clock synchronization in the networked MFFA was initially implemented to synchronize the clocks within the feeding and finishing modules using only the network. Due to delays in communications over the network, this type of clock synchronization is no longer sufficient. Network delay is particularly a problem when more than several feeding and/or finishing modules are configured in the document processing system. In other words, as more feeding and finishing modules are added to the system, the delay over the network increases, causing the synchronization of the

module clocks with the master clock to be unpredictable, thereby unreliable, and often with unacceptable error. Additionally, variability of network delay occurred due to fluctuations in network traffic as the network was used for communications other than clock synchronization.

5 In the networked MFFA, the clock synchronization algorithm was typically designed with the feeding and finishing modules requesting the current master clock time and master module responding to the request, both via the network. More specifically, the basic algorithm went as follows: 1) First, a slave module (i.e., feeding or finishing module) read and saved the current value of its clock. 2) The slave module  
10 then sent a request, over the network, to the master module requesting a read of the master clock. 3) The master module then read its clock and sent the time value back to the slave module over the network. 4) On receipt of the message from the master module, the slave again read the value of its clock. 5) The slave module then subtracted the two readings it had done of its clock and divided the result by two (2) to get one-half  
15 of the delay over the network. 6) The slave then added the result (one-half the delay) to the master clock reading sent by the master module. 7) The slave then compared the adjusted master clock time to the last reading it had done of its clock. The comparison determined whether there was error between the two clocks. 8) Finally, the slave module then adjusted its clock appropriately to account for the error.

20 The problem with this algorithm is that the communications delay over the network is not exactly the same for both directions. Sometimes the request takes longer than the reply and other times the reply takes longer than the request. All the slave can measure is the total time between when the request is made and when the value from the master is received. Unfortunately, the master clock is read sometime in  
25 between when the request is made and when the slave module receives the reply. Accordingly, any algorithm that is based solely on network communications will not be accurate because of the inherent variability of network delay. This variability only gets worse when additional feeding and/or finishing modules communicate over the network.

30 The present invention contemplates a new and improved method for clock synchronization in a networked MFFA configuration of a document processing system that overcomes the above-referenced problems and others.

### Summary of the Invention

In accordance with one aspect of the present invention, a document processing system is provided. The document processing system is comprised of a controller, including a master clock and logic for generating a discrete clock synchronization interrupt signal; a resource, including a slave clock related to operational timing of the resource and circuitry for receiving and processing the discrete interrupt signal; and a control bus, interconnecting the resource and the controller, for distributing the discrete interrupt signal.

10 In accordance with a more limited aspect of the present invention, a method is provided for initially synchronizing the slave clock with the master clock of the document processing system. The method comprising the steps of: a) saving a value of the master clock in the controller; b) generating a discrete clock synchronization interrupt signal in the controller and distributing the discrete interrupt  
15 signal to the resource via the control bus; c) receiving the discrete interrupt signal at the resource and saving a first value of the slave clock; d) sending a message from the resource to the controller via a network to request the value saved for the master clock; e) sending the value saved for the master clock from the controller to the resource via the network; f) receiving the value saved for the master clock at the resource; g) saving  
20 a second value of the slave clock in the resource; h) subtracting the first value from the second value to determine a slave clock difference value; and i) adding the difference value to the value saved for the master clock to determine a synchronized value for the slave clock and setting the slave clock to the synchronized value.

In accordance with a more limited aspect of the present invention, a  
25 method is provided for synchronizing the slave clock with the master clock during steady state operation of the document processing system. The method comprising the steps of: a) saving a value of the master clock in the controller; b) generating a discrete clock synchronization interrupt signal in the controller and distributing the discrete interrupt signal to the resource via the control bus; c) receiving the discrete interrupt  
30 signal at the resource and saving a value of the slave clock; d) sending a message from the resource to the controller via a network to request the value saved for the master

- clock; e) sending the value saved for the master clock from the controller to the resource via the network; f) receiving the value saved for the master clock at the resource; and g) subtracting the value saved for the slave clock from the value saved for the master clock to determine an error value between the slave clock and the master clock and using the error value in an adjustment algorithm to adjust the slave clock to be synchronized with the master clock.

One advantage of the present invention is that it allows the clocks in resources or slave modules (i.e., feeding or finishing modules) of a document processing system configured in a networked MFFA to be more accurately and precisely synchronized with a master clock within the master module (i.e., print engine or controller) of the system.

Another advantage of the present invention is that the accuracy and precision of clock synchronization in a document processing system configured in a networked environment is independent of the quantity of feeding and finishing modules attached to the network nor is it affected by changes in the level of network traffic.

Another advantage of the present invention is that it supports hard real-time service in a document processing system configured in a networked MFFA.

Another advantage of the present invention is that it can synchronize the slave clock of a module or resource to within one (1) clock cycle of the master clock of the controller in the document processing system.

Still further advantages and benefits of the present invention will become apparent to those of ordinary skill in the art upon reading and understanding the following detailed description of the preferred embodiments.

### **Brief Description of the Drawings**

The invention may take form in various components and arrangements of components, and in various steps and arrangements of steps. The drawings are only for purposes of illustrating preferred embodiments and are not to be construed as limiting the invention.

FIGURE 1 is a block diagram of a document processing system incorporating the present invention;

FIGURE 2 is an electrical interconnection diagram of the document processing system of FIGURE 1;

FIGURE 3 is a more detailed electrical interconnection diagram of the document processing system of FIGURE 1;

FIGURE 4 is an electrical interconnection diagram showing the clock synchronization interrupt signal in accordance with the present invention;

FIGURE 5 is a flow chart of a first algorithm for synchronization of real-time clocks in a document processing system in accordance with the present invention; and

FIGURE 6 is a flow chart of a second algorithm for synchronization of real-time clocks in a document processing system in accordance with the present invention.

### **Detailed Description of the Preferred Embodiments**

In describing the present invention, the following terms have been used:

"Real-time" refers to a type of system where system correctness depends not only on outputs, but the timeliness of those outputs. Failure to meet one or more deadlines can result in system failure.

"Hard real-time service" refers to performance guarantees in a real-time system in which missing even one deadline results in system failure.

"Soft real-time service" refers to performance guarantees in a real-time system in which failure to meet deadlines results in performance degradation but not necessarily system failure.

5 "Local area network (LAN)" refers to a network of interconnected computers or computer-controlled devices (i.e., nodes) and connection devices (e.g., switches and routers) that spans a relatively small area at a single site. Each node has its own CPU which controls local operations and communications with other nodes on the LAN. LAN connections are through direct cables (e.g., unshielded twisted pair, coaxial cable, or optical fiber). One of the most common LAN standards is referred to as Ethernet.

10 "Ethernet" refers to a LAN protocol developed by Xerox in cooperation with Digital Equipment Corporation (DEC) and Intel in 1976. Ethernet uses a bus or star topology and supports data transfer rates of 10 megabits per second (Mbps). The Ethernet specification served as the basis for the IEEE 802.3 standard, which specifies the physical and lower software layers.

15 "10 Base T" refers to a type of cable used to connect nodes on an Ethernet network. 10 refers to the transfer rate used on standard Ethernet, 10 Mbps. Base means that the network uses baseband communication rather than broadband communications; T stands for twisted pair. The 10 Base T standard uses a twisted-pair cable with maximum lengths of 100 meters.

20 A document processing system can be comprised of multiple modules or resources (i.e., feeding and finishing devices) attached to a print engine. The modules can be added or removed by the user to provide the feeding and/or finishing capabilities they require. The modules must be accurately and precisely synchronized with a master module (i.e., print engine) in order to correctly perform the printing function. The present invention describes a method that allows the control systems within multiple slave modules or resources of a document processing system to be accurately and  
25 precisely synchronized with a master clock contained in the master module in order to provide certain performance guarantees. The nature of the performance guarantees that the document processing system must provide can be classified into two broad categories: (1) hard real-time guarantees, when the performance requirements of the system will never be violated and (2) soft real-time guarantees, when the requirements  
30 will occasionally, but infrequently, violate the performance requirements.

When providing hard real-time service, the document processing system guarantees apply irrespective of the system functions required to operate simultaneously in order to perform an operation. Consequently, when hard real-time guarantees are required, clock synchronization must be based on the worst-case performance conditions that could be encountered. With soft-real time service, the document processing system cannot guarantee that the performance requirements will be continuously satisfied. Nevertheless, since soft real-time services avoid the worst-case design necessary for hard real-time services, soft real-time services result in a much higher utilization of system resources than hard real-time services. However, the key to the successful deployment of soft real-time services on document processing systems will depend on the effectiveness of the utilization versus performance tradeoffs, which must be conservative enough to ensure that violations of system performance requirements are infrequent, while at the same time ensuring that the system performance is sufficient to make it economically competitive. Depending on the performance required by a particular customer or for a particular operation of the document processing system, the present invention can be implemented for hard real-time service or soft real-time service. Generally, document processing systems are presumed to require hard real-time service. The present invention has been demonstrated to synchronize real-time clocks in slave modules or resources to within one (1) clock cycle of the master clock in the print engine of the document processing system.

With reference to FIGURE 1, a document processing system in accordance with the present invention comprises a marking engine or printer with multiple attached feeding and/or finishing devices. In particular, a marking engine or printer 10 including a user interface with screen 11 is interconnected to multiple document feeding devices 12 and 14 and multiple document finishing devices 16, 18, and 20. As is well known, the feeding devices are sources of printable media like paper for providing the printer 10 with stock for completion of the printing process. Feeding devices are paper trays, and the need for feeding devices is to supply an increased level of printable stock selection to the printer. This could be for the purpose of having multiple colors of tabs or separating stock or front and rear cover stock with a clear coversheet. Or it could be a check printing application where there is a variety of scenes



preprinted and each is fed in a collated order to build a check book with the proper cover sheets and rear cover stock. The finishing devices can be any suitable devices such as sorters, compilers, staplers, folders, or trimmers. It should be noted that FIGURE 1 is only one embodiment and meant to illustrate a functional view of the devices rather than actual physical placement.

The feeding and finishing devices (12, 14, 16, 18, 20) may be referred to collectively as resources with respect to the marking engine or printer 10 (i.e., print engine) of the document processing system. The resources are physically attached to the print engine 10 such that sheets can be fed into the print engine 10 from the feeding devices (12, 14) and sheets or sets of sheets can be transferred from the print engine 10 to the finishing devices (16, 18, 20). The resources are attached to each other such that sheets or sets of sheets can be transferred from one device to another. It should be understood that the mechanical specifications of how these resources are connected to the print engine 10 or to one another are not part of the present invention.

With reference to FIGURE 2, the basic electrical interconnections of the document processing system of FIGURE 1 are shown. The electrical interconnections depicted in FIGURE 2 include a system network 34 (e.g., LAN) and a control bus 36. It should be noted that the architecture of FIGURE 2 is modular and can include any combination of resources. The architecture allows expansion to include additional resources, while less resources than the configuration of five (5) shown in FIGURE 2 are also permitted under the present invention. The preferred embodiment of the present invention uses a 10 Base T Ethernet network with nodes connected in a star topology. However, alternate topologies, alternate cabling standards, and alternate network protocol standards that are able to implement synchronization of real-time clocks as described below are also contemplated.

Continuing to refer to FIGURE 2, printed wiring board assembly (PWBA) 26 of feeding device 12 is connected to the system network 34 and the control bus 36. Likewise, PWBA 24 of feeding device 14, controller PWBA 22 of the print engine 10, PWBA 28 of finisher device 16, PWBA 30 of finisher device 18, and PWBA 32 of finisher device 20 are also connected to the system network 34 and the control bus 36. Controller PWBA 22 is also commonly referred to as a controller. Each resource



detail. More particularly, the controller PWBA 22 includes a master clock 48 related to the operation and timing of the document processing system 10 and a CPU 50 or other circuitry for generating a discrete clock synchronization interrupt signal 40. In addition, the CPU 50 in the controller PWBA 22 responds to the operational and timing data of the master clock 48 to operate in accordance with the operation and timing requirements of the document processing system. Each resource PWBA (24, 26, 28, 30) includes a slave clock 42 related to the operation and timing of the related resource (12, 14, 16, 18) and a CPU 44 or other circuitry for responding to the discrete clock synchronization interrupt signal 40 from the controller PWBA 22. As described below, an object of the present invention is to synchronize the slave clocks 42 of the resources (12, 14, 16, 18) with the master clock 48 of the controller PWBA 22. More generally, the clock synchronization interrupt signal 40 is used to synchronize operations of all the resources (12, 14, 16, 18) with the controller 22, thereby guaranteeing proper real-time operation of the document processing system.

The clock synchronization interrupt signal 40 is one of the discrete control signals distributed via the control bus 36. More particularly, the discrete clock synchronization interrupt signal 40 is distributed from the controller PWBA 22 to the resource PWBA (24, 26, 28, 30). Each resource must provide a one to one interconnect for the discrete control bus signals from where the signals enter the device and where they exit the device. With reference to FIGURE 4, for a typical resource PWBA, the discrete clock synchronization interrupt signal 40 is optically isolated between the PWBA (24, 26, 28, 30) and the controller PWBA 22 by an optical buffer

With reference to FIGURE 5, the present invention provides a first algorithm for synchronization of the slave clock 42 within the resources (12, 14, 16, 18) with the master clock 48 in the controller 22 of the document processing system. This first algorithm is performed one time during initialization of the document processing system. On startup, the controller starts a periodic timer 505. The periodic timer establishes a repetitive interval for a clock synchronization cycle (e.g., two (2) minutes) within the document processing system during steady state operations. When the periodic timer expires 510, the controller reads the current time of its master clock 515

and saves the snapshot time 520. The saved value for the master clock is referred to as Master\_Clock\_Snapshot\_Time. Immediately after reading and saving the master clock snapshot time, the controller sends a clock synchronization interrupt signal to the feeding/finishing resources 525.

- 5           The following steps are performed by each resource (12, 14, 16, 18) configured in the document processing system upon receipt of the first clock synchronization interrupt signal 530 after startup. After receiving the discrete interrupt, the resource immediately reads the current time of its slave clock 535 and saves the snapshot time 540. The saved value is referred to as Slave\_Clock\_Snapshot\_Time\_1.
- 10       Next, the resource sends a message, using the network, to the controller requesting the Master\_Clock\_Snapshot\_Time 545 to be transmitted to it via the network.

- The following steps are performed by the controller 22 each time it receives a request for the last saved Master\_Clock\_Snapshot\_Time 550. First, the controller retrieves the last saved Master\_Clock\_Snapshot\_Time 555. Then, the controller sends the Master\_Clock\_Snapshot\_Time to the requesting resource over the network 560.
- 15

- The following steps are performed by each resource (12, 14, 16, 18) the first time the resource receives the Master\_Clock\_Snapshot\_Time from the controller 565 after startup. First, the resource reads the current time on its slave clock 570 and saves the snapshot time. The saved value is referred to as Slave\_Clock\_Snapshot\_Time\_2. Next, the resource determines the difference between Slave\_Clock\_Snapshot\_Time\_1 and Slave\_Clock\_Snapshot\_Time\_2 575 by subtracting the first snapshot time from the second. The result is the time elapsed since the discrete clock synchronization interrupt was received. The resource then adds the difference to the Master\_Clock\_Snapshot\_Time 580 to determine a synchronized value. Finally, the resource transfer this result (i.e., the Master\_Clock\_Snapshot\_Time adjusted for elapsed time) to the slave clock 585 and the slave clock is set to the synchronized value. At this point the slave clock 42 is initially synchronized with the master clock 48.
- 20
- 25

- With reference to FIGURE 6, the present invention provides a second algorithm for synchronization of the slave clock 42 within the resources (12, 14, 16, 18) with the master clock 48 in the controller 22 of the document processing system. The
- 30

second algorithm is performed after the initialization algorithm of FIGURE 5 and periodically during steady state operations according to the clock synchronization cycle (e.g., two (2) minutes) established by the controller's periodic timer. Each time the periodic timer expires 610, the controller reads the current time of its master clock 615 and saves the snapshot time 620. The saved value for the master clock is referred to as Master\_Clock\_Snapshot\_Time. Immediately after reading and saving the master clock snapshot time, the controller sends a clock synchronization interrupt signal to the feeding/finishing resources 625.

The following steps are performed by each resource (12, 14, 16, 18) configured in the document processing system upon receipt of the clock synchronization interrupt signal 630. After receiving the discrete interrupt, the resource immediately reads the current time of its slave clock 635 and saves the snapshot time 640. The saved value is referred to as Slave\_Clock\_Snapshot\_Time. Next, the resource sends a message, using the network, to the controller requesting the Master\_Clock\_Snapshot\_Time 645 to be transmitted to it via the network.

The following steps are performed by the controller 22 each time it receives a request for the last saved Master\_Clock\_Snapshot\_Time 650. First, the controller retrieves the last saved Master\_Clock\_Snapshot\_Time 655. Then, the controller sends the Master\_Clock\_Snapshot\_Time to the requesting resource over the network 660.

The following steps are performed by each resource (12, 14, 16, 18) each time the resource receives the Master\_Clock\_Snapshot\_Time from the controller 665 during steady state operations. First, the resource determines the difference between Master\_Clock\_Snapshot\_Time and the Slave\_Clock\_Snapshot\_Time 690 by subtracting the slave clock time from the master clock time. The result is the error between the slave clock and the master clock when the clock synchronization interrupt was received. The resource then applies an algorithm to gradually adjust the slave clock for the error 695. Once the adjustment algorithm has eliminated the error or reduced it to acceptable limits, the slave clock 42 is synchronized with the master clock 48.

The present invention enables the networked architecture of Platteter '775 to be implemented in document processing systems that provide real-time service,

